IE 6318 Data Mining and Analytics

Homework 3

Classification Using Bayesian Decision Theory

Submitted by

Sarika Kshatriya

UTA ID:1001483313

[Sarika.kshatriya@mavs.uta.edu](mailto:Sarika.kshatriya@mavs.uta.edu)

1. Bayesian Classification:

With 5-fold cross validation: Assumed the four features are independent and followed normal distribution.

**Matlab Code:**

Classification Using Bayesian main program:

%With 5-fold cross validation: Assumed the four features are independent and followed normal distribution.

clear all;close all;clc

load Iris.csv;

class= [1,2,3];

Y =[class];

X = Iris (:,1:4);

Y = Iris (:,5);

N=5;

AccuracyMatrix = []; confusionMatrix = [];

% 5 fold

%Divide the data into 5 sets in each class

class = unique(Y);

for i = 1:size(class,1)

NoOfFolds = [];

temp = X(Y==class(i),:);

index = find(Y==class(i));

c = numel(index);

%Shuffle the dataset randomly.

rng('shuffle');

temp = temp(randperm(c),:);

quo = floor(c/N);

rem = mod(c, N);

rng('shuffle');

extra = randsample(1:N,rem);

NoOfFolds = ones(N,1) \* quo;

NoOfFolds(extra) = NoOfFolds(extra) + 1;

%Cumulative Sum

foldSizeCumSum = cumsum(NoOfFolds);

%Split the dataset into 5 groups

for j = 1:N

if j==1

eval(['nfold.class' , num2str(i) , '.fold', num2str(j) , ...

'=temp(', num2str(1), ':', num2str(foldSizeCumSum(1)), ',:);']);

else

eval(['nfold.class' , num2str(i), '.fold', num2str(j) , ...

'=temp(', num2str(foldSizeCumSum(j-1)+1), ':' , num2str(foldSizeCumSum(j)), ',:);']);

end

end

end

%For each unique group:

%Take the group as a hold out or test data set

%Take the remaining groups as a training data set

for i = 1:N

testData = []; testLabel = []; trainData = []; trainDataLabel = [];

%test data

for j = 1:size(class,1)

eval(['testData = [testData; nfold.class' , num2str(j), '.fold', num2str(i), '];']);

eval(['testLabel = [testLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(i), ',1),1)];']);

end

%train data

for j = 1:size(class,1)

for h = 1:N

if h ~= i

eval(['trainData = [trainData; nfold.class' , num2str(j), '.fold', num2str(h), '];']);

eval(['trainDataLabel = [trainDataLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(h), ',1),1)];']);

end

end

end

%-----------------------------------------------------

bays = fitcnb(trainData, trainDataLabel);

predData = predict(bays,testData);

%disp(PredData);

%Accuracy

accuracy = sum(predData == testLabel)/length(testLabel);

% accuracyMatrix = [accuracyMatrix; accuracy];

%AccuracyMatrix = [];

disp("For fold "+i);

stats=confusionmatStats(predData,testLabel);

disp("--------------------------------------------------------------------------")

%disp("Accuracy :+ accuracy);

%Calculate Confusion-matrix

%confusionMatrix = confusionmat (testLabel,predData);

%disp("Confusion-matrix for fold :"+i+" is:");

%disp(confusionMatrix);

%figure;

%confusionchart(confusionMatrix);

eval(['confusionMatrix.fold', num2str(i), ' = confusionmat(testLabel, predData);']);

if i == 1

confusionMatrix.allFolds = confusionMatrix.fold1;

else

eval(['confusionMatrix.allFolds = confusionMatrix.allFolds + confusionMatrix.fold', num2str(i),';']);

end

AccuracyMatrix = [AccuracyMatrix; accuracy];

end

%----------------------------------------------------

disp("For all folds-");

disp("Mean Accuracy :");

disp(mean(AccuracyMatrix));

disp("Confusion Matrix:");

disp(confusionMatrix.allFolds);

confusionchart(confusionMatrix.allFolds);

%----------------------------------------------------

numOfClasses = 3;

totalSamples = 150;

value1=confusionMatrix.allFolds;

[TP,TN,FP,FN,accuracy,sensitivity,specificity] = deal(zeros(numOfClasses,1));

for class = 1:numOfClasses

TP(class) = value1(class,class);

tempMat = value1;

tempMat(:,class) = []; % remove column

tempMat(class,:) = []; % remove row

TN(class) = sum(sum(tempMat));

FP(class) = sum(value1(:,class))-TP(class);

FN(class) = sum(value1(class,:))-TP(class);

end

for class = 1:numOfClasses

accuracy(class) = (TP(class) + TN(class)) / totalSamples;

sensitivity(class) = TP(class) / (TP(class) + FN(class));

specificity(class) = TN(class) / (FP(class) + TN(class));

end

disp("For each class:");

field2 = 'accuracy';

value2 = accuracy;

disp(field2);disp(value2);

field3 = 'sensitivity';

value3 = sensitivity;

disp(field3);disp(value3);

field4 = 'specificity';

value4 = specificity;

disp(field4);disp(value4);

%----------------------------------------------------

ConfusionmatStats function:

function stats = confusionmatStats(group,grouphat)

% INPUT

% group = true class labels

% grouphat = predicted class labels

%

% OR INPUT

% stats = confusionmatStats(group);

% group = confusion matrix from matlab function (confusionmat)

%

% OUTPUT

% stats is a structure array

% stats.confusionMat

% Predicted Classes

% p' n'

% \_\_\_|\_\_\_\_\_|\_\_\_\_\_|

% Actual p | | |

% Classes n | | |

%

%

% TP: true positive, TN: true negative,

% FP: false positive, FN: false negative

%%----------------------------------------------------

field1 = 'confusion\_Matrix';

if nargin < 2

value1 = group;

else

[value1,gorder] = confusionmat(group,grouphat);

end

disp(field1);

C=confusionmat(group,grouphat);

disp(C);

figure

confusionchart(C);

numOfClasses = size(value1,1);

totalSamples = sum(sum(value1));

[TP,TN,FP,FN,accuracy,sensitivity,specificity] = deal(zeros(numOfClasses,1));

for class = 1:numOfClasses

TP(class) = value1(class,class);

tempMat = value1;

tempMat(:,class) = []; % remove column

tempMat(class,:) = []; % remove row

TN(class) = sum(sum(tempMat));

FP(class) = sum(value1(:,class))-TP(class);

FN(class) = sum(value1(class,:))-TP(class);

end

for class = 1:numOfClasses

accuracy(class) = (TP(class) + TN(class)) / totalSamples;

sensitivity(class) = TP(class) / (TP(class) + FN(class));

specificity(class) = TN(class) / (FP(class) + TN(class));

end

disp("Mean Accuracy:");

disp(mean(accuracy));

disp("For each class:");

field2 = 'accuracy';

value2 = accuracy;

disp(field2);

disp(value2);

field3 = 'sensitivity';

value3 = sensitivity;

disp(field3);

disp(value3);

field4 = 'specificity';

value4 = specificity;

disp(field4);

disp(value4);

stats = struct(field1,value1,field2,value2,field3,value3,field4,value4);

end

%----------------------------------------------------

**Output of above program:**

For fold 1

confusion\_Matrix

10 0 0

0 10 2

0 0 8

For each class:

accuracy

1.0000

0.9333

0.9333

sensitivity

1.0000

0.8333

1.0000

specificity

1.0000

1.0000

0.9091

--------------------------------------------------------------------------

For fold 2

confusion\_Matrix

10 0 0

0 8 1

0 2 9

For each class:

accuracy

1.0000

0.9000

0.9000

sensitivity

1.0000

0.8889

0.8182

specificity

1.0000

0.9048

0.9474

--------------------------------------------------------------------------

For fold 3

confusion\_Matrix

10 0 0

0 10 1

0 0 9

For each class:

accuracy

1.0000

0.9667

0.9667

sensitivity

1.0000

0.9091

1.0000

specificity

1.0000

1.0000

0.9524

--------------------------------------------------------------------------

For fold 4

confusion\_Matrix

10 0 0

0 9 0

0 1 10

For each class:

accuracy

1.0000

0.9667

0.9667

sensitivity

1.0000

1.0000

0.9091

specificity

1.0000

0.9524

1.0000

--------------------------------------------------------------------------

For fold 5

confusion\_Matrix

10 0 0

0 10 0

0 0 10

For each class:

accuracy

1

1

1

sensitivity

1

1

1

specificity

1

1

1

For all folds-

Confusion Matrix:

50 0 0

0 47 3

0 4 46

Mean Accuracy:

0.9689

For each class:

accuracy

1.0000

0.9533

0.9533

sensitivity

1.0000

0.9400

0.9200

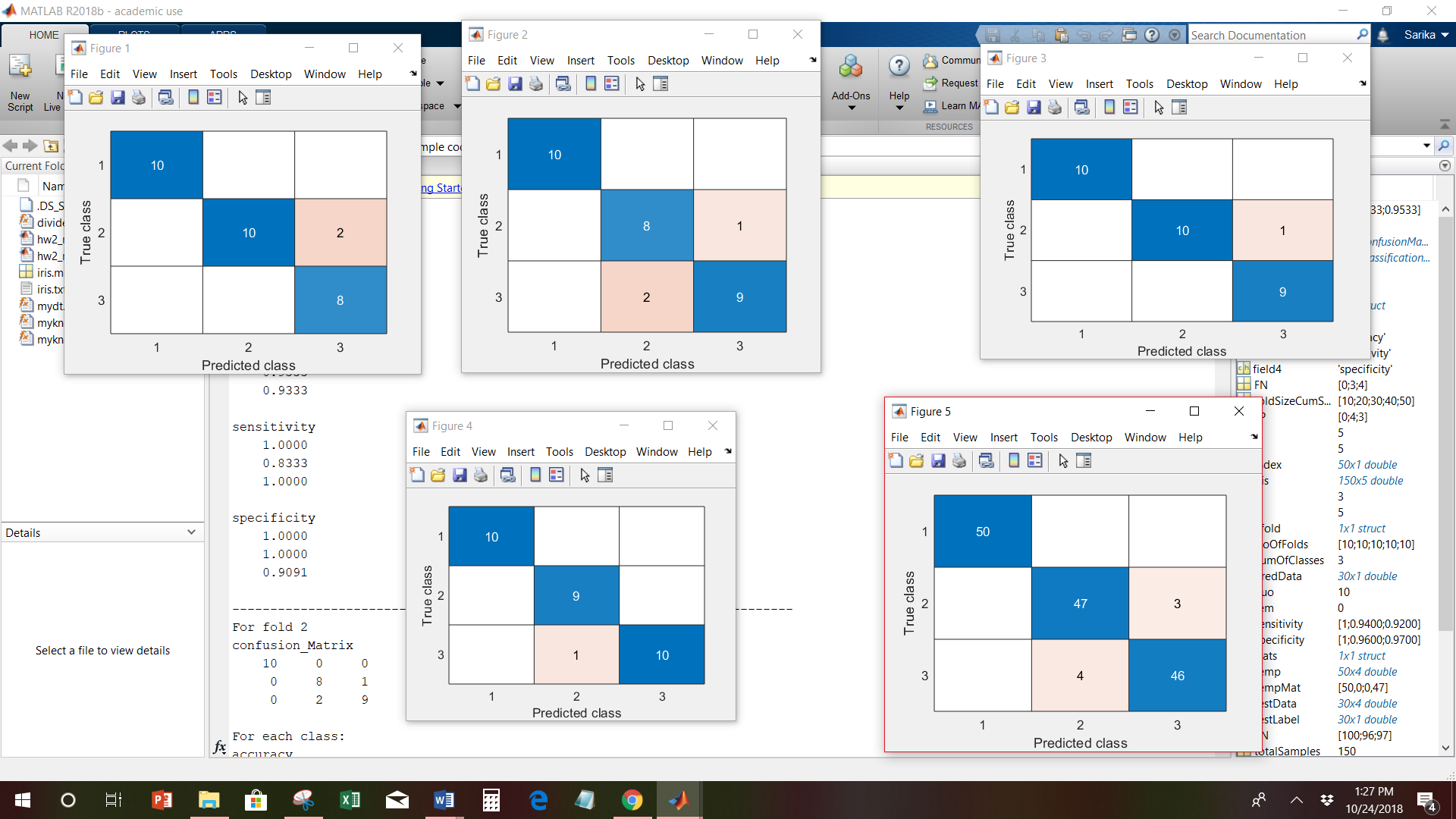
specificity

1.0000

0.9600

0.9700

Confusion Matrix Chart:



2) Multivariate normal distribution.

**Matlab Code:**

Classification Multivariate normal distribution :

%With 5-fold cross validation: followed multivariate normal distribution.

clear all;

close all;

clc

load Iris.csv;

class= [1,2,3];

Y =[class];

X = Iris (:,1:4);

Y = Iris (:,5);

N=5;

AccuracyMatrix = []; confusionMatrix = [];

% 5 fold

%Divide the data into 5 sets in each class

class = unique(Y);

for i = 1:size(class,1)

NoOfFolds = [];

temp = X(Y==class(i),:);

index = find(Y==class(i));

c = numel(index);

%Shuffle the dataset randomly.

rng('shuffle');

temp = temp(randperm(c),:);

quo = floor(c/N);

rem = mod(c, N);

rng('shuffle');

extra = randsample(1:N,rem);

NoOfFolds = ones(N,1) \* quo;

NoOfFolds(extra) = NoOfFolds(extra) + 1;

%Cumulative Sum

foldSizeCumSum = cumsum(NoOfFolds);

%Split the dataset into 5 groups

for j = 1:N

if j==1

eval(['nfold.class' , num2str(i) , '.fold', num2str(j) , ...

'=temp(', num2str(1), ':', num2str(foldSizeCumSum(1)), ',:);']);

else

eval(['nfold.class' , num2str(i), '.fold', num2str(j) , ...

'=temp(', num2str(foldSizeCumSum(j-1)+1), ':' , num2str(foldSizeCumSum(j)), ',:);']);

end

end

end

%For each unique group:

%Take the group as a hold out or test data set

%Take the remaining groups as a training data set

for i = 1:N

testData = []; testLabel = []; trainData = []; trainDataLabel = [];

%test data

for j = 1:size(class,1)

eval(['testData = [testData; nfold.class' , num2str(j), '.fold', num2str(i), '];']);

eval(['testLabel = [testLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(i), ',1),1)];']);

end

%train data

for j = 1:size(class,1)

for h = 1:N

if h ~= i

eval(['trainData = [trainData; nfold.class' , num2str(j), '.fold', num2str(h), '];']);

eval(['trainDataLabel = [trainDataLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(h), ',1),1)];']);

end

end

end

%-------------------------------------------------------

predData=myMVFunction(trainData,trainDataLabel,testData);

disp("For fold "+i);

stats=confusionmatStats(predData,testLabel);

disp("--------------------------------------------------------------------------")

eval(['confusionMatrix.fold', num2str(i), ' = confusionmat(testLabel, predData);']);

if i == 1

confusionMatrix.allFolds = confusionMatrix.fold1;

else

eval(['confusionMatrix.allFolds = confusionMatrix.allFolds + confusionMatrix.fold', num2str(i),';']);

end

AccuracyMatrix = [AccuracyMatrix; accuracy];

end

%----------------------------------------------------

disp("For all folds-");

disp("Mean Accuracy :");

disp(mean(AccuracyMatrix));

disp("Confusion Matrix:");

disp(confusionMatrix.allFolds);

confusionchart(confusionMatrix.allFolds);

numOfClasses = 3;

totalSamples = 150;

value1=confusionMatrix.allFolds;

[TP,TN,FP,FN,accuracy,sensitivity,specificity] = deal(zeros(numOfClasses,1));

for class = 1:numOfClasses

TP(class) = value1(class,class);

tempMat = value1;

tempMat(:,class) = []; % remove column

tempMat(class,:) = []; % remove row

TN(class) = sum(sum(tempMat));

FP(class) = sum(value1(:,class))-TP(class);

FN(class) = sum(value1(class,:))-TP(class);

end

for class = 1:numOfClasses

accuracy(class) = (TP(class) + TN(class)) / totalSamples;

sensitivity(class) = TP(class) / (TP(class) + FN(class));

specificity(class) = TN(class) / (FP(class) + TN(class));

end

%----------------------------------------------------

disp("For each class:");

field2 = 'accuracy';

value2 = accuracy;

disp(field2);disp(value2);

field3 = 'sensitivity';

value3 = sensitivity;

disp(field3);disp(value3);

field4 = 'specificity';

value4 = specificity;

disp(field4);disp(value4);

%----------------------------------------------------

ConfusionmatStats function:

function stats = confusionmatStats(group,grouphat)

field1 = 'confusion\_Matrix';

if nargin < 2

value1 = group;

else

[value1,gorder] = confusionmat(group,grouphat);

end

disp(field1);

C=confusionmat(group,grouphat);

disp(C);

figure

confusionchart(C);

numOfClasses = size(value1,1);

totalSamples = sum(sum(value1));

[TP,TN,FP,FN,accuracy,sensitivity,specificity] = deal(zeros(numOfClasses,1));

for class = 1:numOfClasses

TP(class) = value1(class,class);

tempMat = value1;

tempMat(:,class) = []; % remove column

tempMat(class,:) = []; % remove row

TN(class) = sum(sum(tempMat));

FP(class) = sum(value1(:,class))-TP(class);

FN(class) = sum(value1(class,:))-TP(class);

end

for class = 1:numOfClasses

accuracy(class) = (TP(class) + TN(class)) / totalSamples;

sensitivity(class) = TP(class) / (TP(class) + FN(class));

specificity(class) = TN(class) / (FP(class) + TN(class));

end

disp("For each class:");

field2 = 'accuracy';

value2 = accuracy;

disp(field2);disp(value2);

field3 = 'sensitivity';

value3 = sensitivity;

disp(field3);disp(value3);

field4 = 'specificity';

value4 = specificity;

disp(field4);disp(value4);

stats = struct(field1,value1,field2,value2,field3,value3,field4,value4);

end

%--------------------------------------------------

Multivariate Distribution function

function pred = myMVFunction(trainData,trainDataLabel,testData)

class = unique(trainDataLabel);

for c = 1:length(class)

Prior = length(trainDataLabel(trainDataLabel==class(c))) / length(trainDataLabel);

c\_id = find(trainDataLabel == c);

Posterior(:,c) = Prior \* mvnpdf(testData, mean(trainData(c\_id,:)), cov(trainData(c\_id,:)));

end

[num,pred]=max(Posterior,[],2);

end

**Output of above program:**

For fold 1

confusion\_Matrix

10 0 0

0 9 0

0 1 10

For each class:

accuracy

1.0000

0.9667

0.9667

sensitivity

1.0000

1.0000

0.9091

specificity

1.0000

0.9524

1.0000

--------------------------------------------------------------------------

For fold 2

confusion\_Matrix

10 0 0

0 7 0

0 3 10

For each class:

accuracy

1.0000

0.9000

0.9000

sensitivity

1.0000

1.0000

0.7692

specificity

1.0000

0.8696

1.0000

--------------------------------------------------------------------------

For fold 3

confusion\_Matrix

10 0 0

0 10 0

0 0 10

For each class:

accuracy

1

1

1

sensitivity

1

1

1

specificity

1

1

1

--------------------------------------------------------------------------

For fold 4

confusion\_Matrix

10 0 0

0 10 1

0 0 9

For each class:

accuracy

1.0000

0.9667

0.9667

sensitivity

1.0000

0.9091

1.0000

specificity

1.0000

1.0000

0.9524

--------------------------------------------------------------------------

For fold 5

confusion\_Matrix

10 0 0

0 10 0

0 0 10

For each class:

accuracy

1

1

1

sensitivity

1

1

1

specificity

1

1

1

For all folds-

Mean Accuracy :

0.9667

Confusion Matrix:

50 0 0

0 46 4

0 1 49

For each class:

accuracy

1.0000

0.9667

0.9667

sensitivity

1.0000

0.9200

0.9800

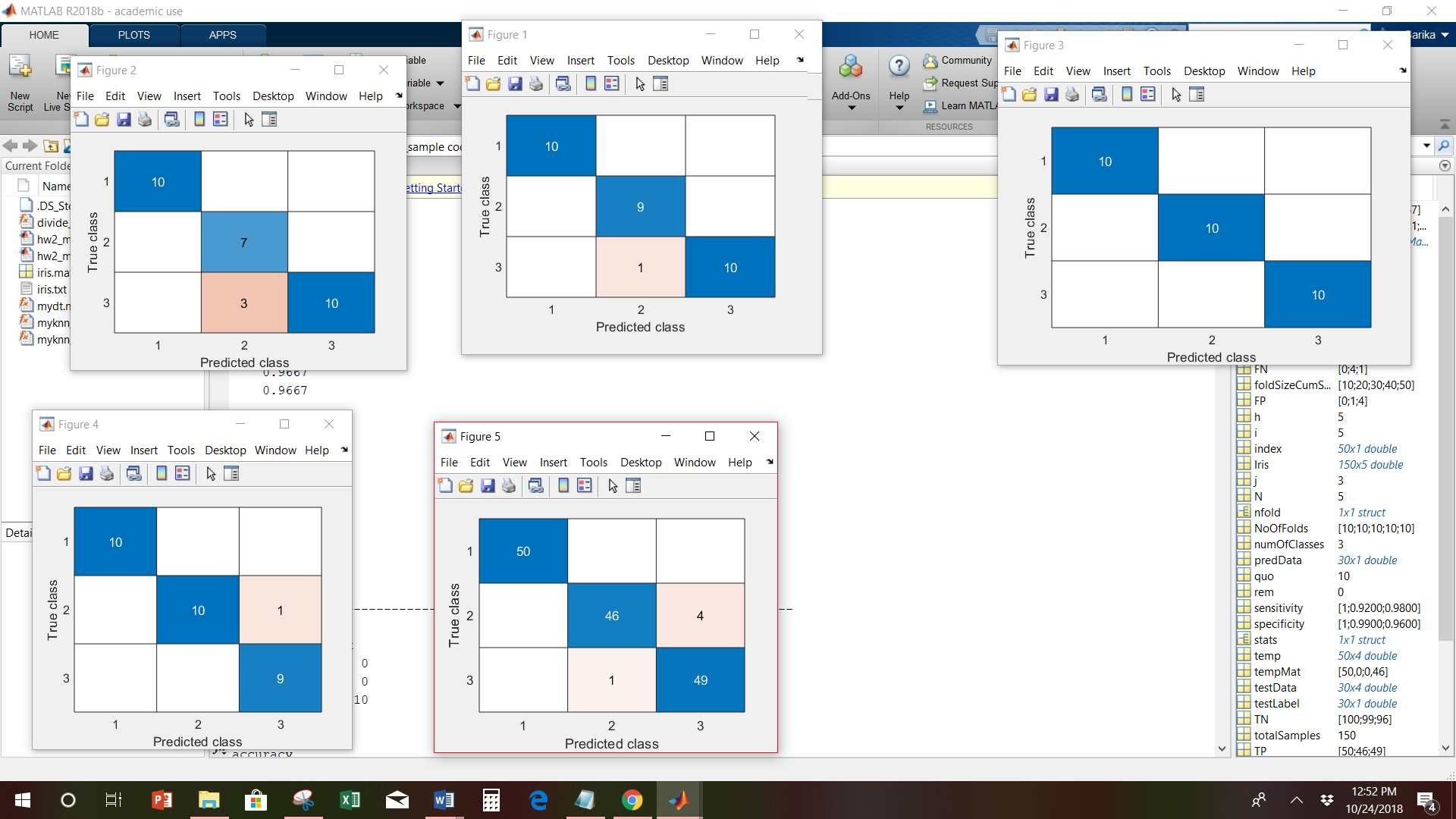
specificity

1.0000

0.9900

0.9600

Confusion Matrix chart for all folds:



2. Discriminant function:

used discriminant function :



1. Bayesian classification function :

Used 5-fold cross-validation

**Matlab Code:**

Classification discriminant function main program:

%With 5-fold cross validation: with discriminant function.

clear all;close all;clc

load Iris.csv;

class= [1,2,3];

Y =[class];

X = Iris (:,1:4);

Y = Iris (:,5);

N=5;

AccuracyMatrix = []; confusionMatrix = [];

% 5 fold

%Divide the data into 5 sets in each class

class = unique(Y);

for i = 1:size(class,1)

NoOfFolds = [];

temp = X(Y==class(i),:);

index = find(Y==class(i));

c = numel(index);

%Shuffle the dataset randomly.

rng('shuffle');

temp = temp(randperm(c),:);

quo = floor(c/N);

rem = mod(c, N);

rng('shuffle');

extra = randsample(1:N,rem);

NoOfFolds = ones(N,1) \* quo;

NoOfFolds(extra) = NoOfFolds(extra) + 1;

%Cumulative Sum

foldSizeCumSum = cumsum(NoOfFolds);

%Split the dataset into 5 groups

for j = 1:N

if j==1

eval(['nfold.class' , num2str(i) , '.fold', num2str(j) , ...

'=temp(', num2str(1), ':', num2str(foldSizeCumSum(1)), ',:);']);

else

eval(['nfold.class' , num2str(i), '.fold', num2str(j) , ...

'=temp(', num2str(foldSizeCumSum(j-1)+1), ':' , num2str(foldSizeCumSum(j)), ',:);']);

end

end

end

%For each unique group:

%Take the group as a hold out or test data set

%Take the remaining groups as a training data set

for i = 1:N

testData = []; testLabel = []; trainData = []; trainDataLabel = [];

%test data

for j = 1:size(class,1)

eval(['testData = [testData; nfold.class' , num2str(j), '.fold', num2str(i), '];']);

eval(['testLabel = [testLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(i), ',1),1)];']);

end

%train data

for j = 1:size(class,1)

for h = 1:N

if h ~= i

eval(['trainData = [trainData; nfold.class' , num2str(j), '.fold', num2str(h), '];']);

eval(['trainDataLabel = [trainDataLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(h), ',1),1)];']);

end

end

end

%---------------------------------------------------------

%call discriminant function

predData= myDisc(trainData,trainDataLabel,testData);

disp("For fold "+i);

stats=confusionmatStats(predData,testLabel);

disp("--------------------------------------------------------------------------")

eval(['confusionMatrix.fold', num2str(i), ' = confusionmat(testLabel, predData);']);

if i == 1

confusionMatrix.allFolds = confusionMatrix.fold1;

else

eval(['confusionMatrix.allFolds = confusionMatrix.allFolds + confusionMatrix.fold', num2str(i),';']);

end

end

%---------------------------------------------------------

disp("For all folds-");

disp("Confusion Matrix:");

disp(confusionMatrix.allFolds);

confusionchart(confusionMatrix.allFolds);

numOfClasses = 3;

totalSamples = 150;

value1=confusionMatrix.allFolds;

[TP,TN,FP,FN,accuracy,sensitivity,specificity] = deal(zeros(numOfClasses,1));

for class = 1:numOfClasses

TP(class) = value1(class,class);

tempMat = value1;

tempMat(:,class) = []; % remove column

tempMat(class,:) = []; % remove row

TN(class) = sum(sum(tempMat));

FP(class) = sum(value1(:,class))-TP(class);

FN(class) = sum(value1(class,:))-TP(class);

end

for class = 1:numOfClasses

accuracy(class) = (TP(class) + TN(class)) / totalSamples;

end

disp("For each class:");

field2 = 'accuracy';

value2 = accuracy;

disp(field2);

disp(value2);

disp("Mean Accuracy:");

disp(mean(accuracy));

%---------------------------------------------------------

ConfusionmatStats function:

function stats = confusionmatStats(group,grouphat)

field1 = 'confusion\_Matrix';

if nargin < 2

value1 = group;

else

[value1,gorder] = confusionmat(group,grouphat);

end

disp(field1);

C=confusionmat(group,grouphat);

disp(C);

numOfClasses = size(value1,1);

totalSamples = sum(sum(value1));

[TP,TN,FP,FN,accuracy,sensitivity,specificity] = deal(zeros(numOfClasses,1));

for class = 1:numOfClasses

TP(class) = value1(class,class);

tempMat = value1;

tempMat(:,class) = []; % remove column

tempMat(class,:) = []; % remove row

TN(class) = sum(sum(tempMat));

FP(class) = sum(value1(:,class))-TP(class);

FN(class) = sum(value1(class,:))-TP(class);

end

for class = 1:numOfClasses

accuracy(class) = (TP(class) + TN(class)) / totalSamples;

sensitivity(class) = TP(class) / (TP(class) + FN(class));

specificity(class) = TN(class) / (FP(class) + TN(class));

end

disp("For each class:");

field2 = 'accuracy';

value2 = accuracy;

disp(field2);

disp(value2);

disp("Mean Accuracy: ");

disp(mean(value2));

stats = struct(field1,value1,field2,value2);

end

%---------------------------------------------------------

Discriminant function:

function Lpred= myDisc(trainData,trainDataLabel,testData)

cl = unique(trainDataLabel);

giX=[];

for c = 1:length(cl)

PWi= length(trainDataLabel(trainDataLabel==cl(c))) / length(trainDataLabel);

UniqueCl= find(trainDataLabel == cl(c));

CovarianceMatrix= cov(trainData(UniqueCl,:));

mu= mean(trainData(UniqueCl,:))';

constW= -0.5\*mu'\*inv(CovarianceMatrix)\*mu - log(det(CovarianceMatrix))/2 + log(PWi);

giX(:,c) = diag(bsxfun(@plus, testData \* -((inv(CovarianceMatrix))/2) \* testData', ((inv(CovarianceMatrix)) \* mu)' \* testData' + constW));

end

[~,Lpred] = max(giX,[],2);

end

**Output of above program:**

For fold 1

confusion\_Matrix

10 0 0

0 9 0

0 1 10

For each class:

accuracy

1.0000

0.9667

0.9667

Mean Accuracy:

0.9778

--------------------------------------------------------------------------

For fold 2

confusion\_Matrix

10 0 0

0 10 0

0 0 10

For each class:

accuracy

1

1

1

Mean Accuracy:

1

--------------------------------------------------------------------------

For fold 3

confusion\_Matrix

10 0 0

0 9 0

0 1 10

For each class:

accuracy

1.0000

0.9667

0.9667

Mean Accuracy:

0.9778

--------------------------------------------------------------------------

For fold 4

confusion\_Matrix

10 0 0

0 10 1

0 0 9

For each class:

accuracy

1.0000

0.9667

0.9667

Mean Accuracy:

0.9778

--------------------------------------------------------------------------

For fold 5

confusion\_Matrix

10 0 0

0 9 0

0 1 10

For each class:

accuracy

1.0000

0.9667

0.9667

Mean Accuracy:

0.9778

--------------------------------------------------------------------------

For all folds-

Confusion Matrix:

50 0 0

0 47 3

0 1 49

For each class:

accuracy

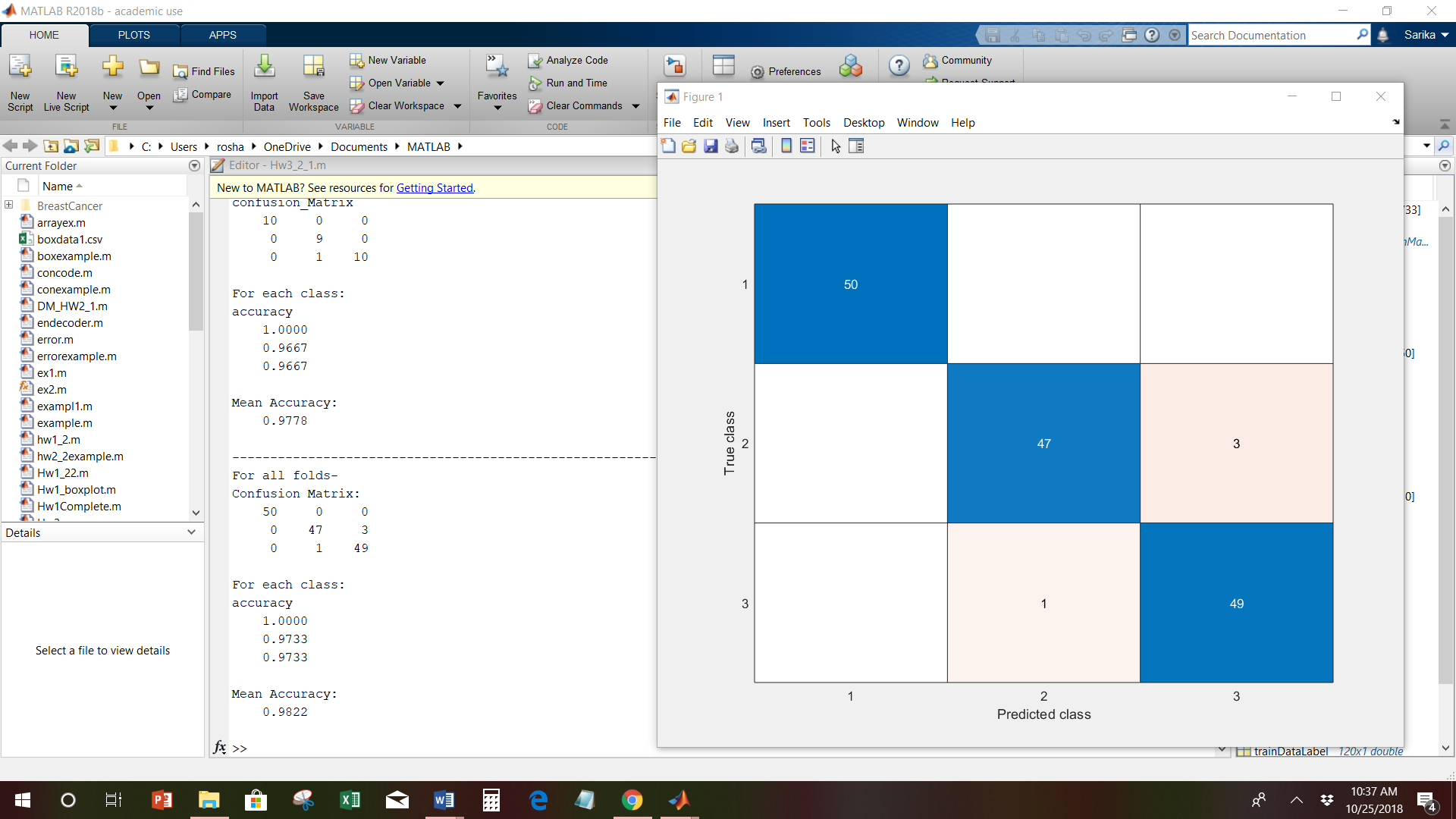
1.0000

0.9733

0.9733

Mean Accuracy:

0.9822



2. Binary classification for the Breast Cancer dataset using the Bayesian classification function.

Done 5-fold cross-validation

The data is downloaded from the UCI Machine Learning Repository: <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra>

**Matlab Code:**

Classification discriminant function main program:

%Binary classification for the Breast Cancer dataset using the Bayesian classification function.

%Done 5-fold cross-validation

%The data is downloaded from the UCI Machine Learning Repository:

%https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra

clear all; close all; clc

load BreastCancer.csv;

class= [1,2];

Y =[class];

X = BreastCancer (:,1:9);

Y = BreastCancer (:,10);

totalSamples=116;

for i=1:116

if Y(i)<2

Y(i)=0;%For healthy ppl

else

Y(i)=1;%for patients

end

end

%----------------------------------------------------------

%assign binary values 0 and 1

N=5;

accuracyMatrix = []; confusionMatrix = [];

% 5 fold

%Divide the data into 5 sets in each class

class = unique(Y);

for i = 1:size(class,1)

NoOfFolds = [];

temp = X(Y==class(i),:);

index = find(Y==class(i));

c = numel(index);

%Shuffle the dataset randomly.

rng('shuffle');

temp = temp(randperm(c),:);

quo = floor(c/N);

rem = mod(c, N);

rng('shuffle');

extra = randsample(1:N,rem);

NoOfFolds = ones(N,1) \* quo;

NoOfFolds(extra) = NoOfFolds(extra) + 1;

%Cumulative Sum

foldSizeCumSum = cumsum(NoOfFolds);

%Split the dataset into 5 groups

for j = 1:N

if j==1

eval(['nfold.class' , num2str(i) , '.fold', num2str(j) , ...

'=temp(', num2str(1), ':', num2str(foldSizeCumSum(1)), ',:);']);

else

eval(['nfold.class' , num2str(i), '.fold', num2str(j) , ...

'=temp(', num2str(foldSizeCumSum(j-1)+1), ':' , num2str(foldSizeCumSum(j)), ',:);']);

end

end

end

%For each unique group:

%Take the group as a hold out or test data set

%Take the remaining groups as a training data set

for i = 1:N

testData = []; testLabel = []; trainData = []; trainDataLabel = [];

%test data

for j = 1:size(class,1)

eval(['testData = [testData; nfold.class' , num2str(j), '.fold', num2str(i), '];']);

eval(['testLabel = [testLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(i), ',1),1)];']);

end

%train data

for j = 1:size(class,1)

for h = 1:N

if h ~= i

eval(['trainData = [trainData; nfold.class' , num2str(j), '.fold', num2str(h), '];']);

eval(['trainDataLabel = [trainDataLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(h), ',1),1)];']);

end

end

end

%-----------------------------------------------------------------------------------

%call discriminant function

predData= myDisc(trainData,trainDataLabel,testData);

disp("For fold "+i);

stats=confusionmatStats(predData,testLabel);

disp("--------------------------------------------------------------------------")

eval(['confusionMatrix.fold', num2str(i), ' = confusionmat(testLabel, predData);']);

if i == 1

confusionMatrix.allFolds = confusionMatrix.fold1;

else

eval(['confusionMatrix.allFolds = confusionMatrix.allFolds + confusionMatrix.fold', num2str(i),';']);

end

end

%---------------------------------------------------------

disp("For all folds-");

disp("Confusion Matrix:");

disp(confusionMatrix.allFolds);

C=confusionMatrix.allFolds;

confusionchart(confusionMatrix.allFolds);

TP=C(1,1);

TN=C(2,2);

acc=(TP+TN)/totalSamples;

disp("Mean Accuracy:"+acc);

%---------------------------------------------------------

Discriminant function:

function Lpred= myDisc(trainData,trainDataLabel,testData)

cl = unique(trainDataLabel);

giX=[];

for c = 1:length(cl)

PWi= length(trainDataLabel(trainDataLabel==cl(c))) / length(trainDataLabel);

UniqueCl= find(trainDataLabel == cl(c));

CovarianceMatrix= cov(trainData(UniqueCl,:));

mu= mean(trainData(UniqueCl,:))';

constW= -0.5\*mu'\*inv(CovarianceMatrix)\*mu - log(det(CovarianceMatrix))/2 + log(PWi);

giX(:,c) = diag(bsxfun(@plus, testData \* -((inv(CovarianceMatrix))/2) \* testData', ((inv(CovarianceMatrix)) \* mu)' \* testData' + constW));

end

[~,Lpred] = max(giX,[],2);

end

%---------------------------------------------------------

ConfusionmatStats function:

function stats = confusionmatStats(group,grouphat)

field1 = 'confusion\_Matrix';

if nargin < 2

value1 = group;

else

[value1,gorder] = confusionmat(group,grouphat);

end

disp(field1);

C=confusionmat(group,grouphat);

disp(C);

numOfClasses = size(value1,1);

totalSamples = sum(sum(value1));

[TP,TN,FP,FN,accuracy,sensitivity,specificity] = deal(zeros(numOfClasses,1));

for class = 1:numOfClasses

TP(class) = value1(class,class);

tempMat = value1;

tempMat(:,class) = []; % remove column

tempMat(class,:) = []; % remove row

TN(class) = sum(sum(tempMat));

FP(class) = sum(value1(:,class))-TP(class);

FN(class) = sum(value1(class,:))-TP(class);

end

for class = 1:numOfClasses

accuracy(class) = (TP(class) + TN(class)) / totalSamples;

sensitivity(class) = TP(class) / (TP(class) + FN(class));

specificity(class) = TN(class) / (FP(class) + TN(class));

end

disp("For each class:");

field2 = 'accuracy';

value2 = accuracy;

disp(field2);

disp(value2);

disp("Mean Accuracy: ");

disp(mean(value2));

stats = struct(field1,value1,field2,value2);

end

**Output of above program:**

For fold 1

confusion\_Matrix

10 7

1 6

For each class:

accuracy

0.6667

0.6667

Mean Accuracy:

0.6667

--------------------------------------------------------------------------

For fold 2

confusion\_Matrix

9 3

1 9

For each class:

accuracy

0.8182

0.8182

Mean Accuracy:

0.8182

--------------------------------------------------------------------------

For fold 3

confusion\_Matrix

9 8

2 5

For each class:

accuracy

0.5833

0.5833

Mean Accuracy:

0.5833

--------------------------------------------------------------------------

For fold 4

confusion\_Matrix

8 8

2 5

For each class:

accuracy

0.5652

0.5652

Mean Accuracy:

0.5652

--------------------------------------------------------------------------

For fold 5

confusion\_Matrix

9 7

1 6

For each class:

accuracy

0.6522

0.6522

Mean Accuracy:

0.6522

--------------------------------------------------------------------------

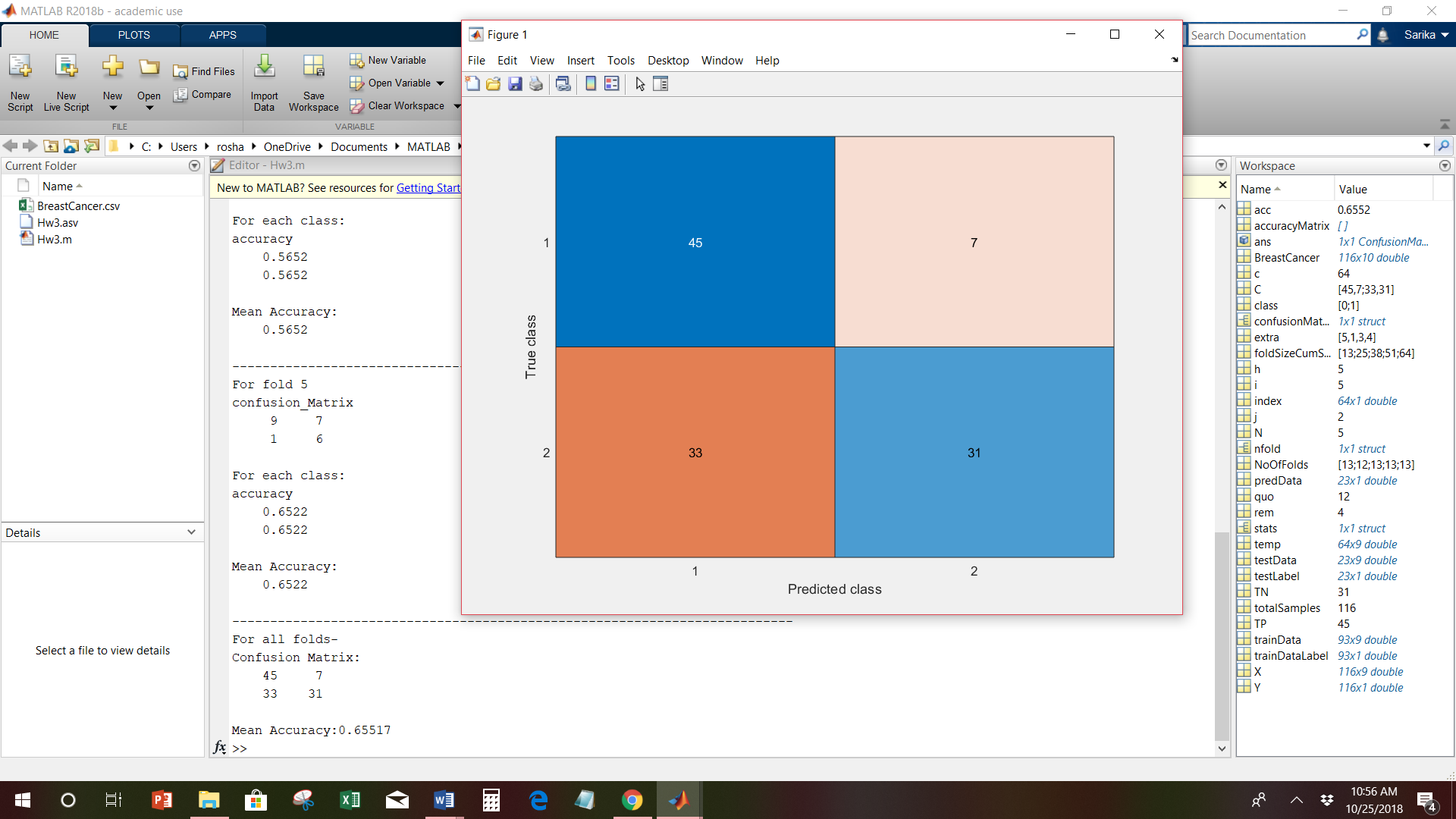
For all folds-

Confusion Matrix:

45 7

33 31

Mean Accuracy:0.65517



References:

<https://in.mathworks.com/help/> <https://in.mathworks.com/help/stats/examples/classification.html>

<https://machinelearningmastery.com/k-fold-cross-validation/>

<https://in.mathworks.com/matlabcentral/fileexchange/46035-confusionmatstats-group-grouphat>

Lecture3\_Classification Basics

Lecture4\_Bayesian Theory

Lecture5\_Linear Discriminant Functions